**INTRODUCTION TO SCRIPTING LANGUAGES**

**Scripts and programs:**

Scripting is the action of writing scripts using a scripting language, distinguishing neatly between programs, which are written in conventional programming language such as C, C++, java, and scripts, which are written using a different kind of language.

We could reasonably argue that the use of scripting languages is just another kind of programming. Scripting languages are used for is qualitatively different from conventional programming languages like C++ and Ada address the problem of developing large applications from the ground up, employing a team of professional programmers, starting from well-defined specifications, and meeting specified performance constraints.

Scripting languages, on other hand, address different problems:

* Building applications from ‘off the shelf’ components
* Controlling applications that have a programmable interface
* Writing programs where speed of development is more important than run-time efficiency.

The most important difference is that scripting languages incorporate features that enhance the productivity of the user in one way or another, making them accessible to people who would not normally describe themselves as programmers, their primary employment being in some other capacity. Scripting languages make programmers of us all, to some extent.

**Origin of scripting:**

All the devices like camera, cell phones, DVD player, and CD player etc uses computer technology. Devices that rely on computer technology are almost everywhere, not only in work place but also in schools, colleges, home etc...

Computer technology plays an important role in

- Entertainment

- Education

- Medicine

- Manufacturing

- Communications

- Government etc...

Scripting languages are used to extend the functionality of existing software, such as creating dynamic content for web pages and automating repetitive tasks.

Developing software requires programming and programming consists of writing a series of detailed instructions that tell a computer what to do. The code for a typical software application can be very complex and long.

In many cases, you may want to do something simpler than developing your own software. For example, let's say you have a few hundred digital photographs and you need to resize them and then upload them to a website. You have photo-editing software that can resize photographs. However, doing this one-by-one is going to be cumbersome, so you want a tool that automates this process. You need a script to do this.

A script is a set of instructions that automates the execution of tasks which would otherwise be done one-by-one by a human operator.

You can write scripts using a scripting language, which is a type of programming language to write scripts. Scripting is a form of programming, but it typically takes place within an existing application.

The use of the word ‘script’ in a computing context dates back to the early 1970s,when the originators of the UNIX operating system create the term ‘shell script’ for sequence ofcommands that were to be read from a file and follow in sequence as if they had been typed in at the keyword. E.g. an ‘AWKscript’, a ‘perl script’ etc.. the name ‘script ‘ being used for a text file that was intended to be executed directly rather than being compiled to a different form of file prior to execution.

Other early occurrences of the term ‘script’ can be found. For example, in a DOS-based system, use of a dial-up connection to a remote system required a communication package that used proprietary language to write scripts to automate the sequence of operations required to establish a connection to a remote system.

Note that if we regard a scripts as a sequence of commands to control an application or a device, a configuration file such as a UNIX ‘make file’ could be regard as a script.

However, scripts only become interesting when they have the added value that comes from using programming concepts such as loops and branches.

**Scripting today:**

SCRIPTING IS USED WITH 3 DIFFRENT MEANINGS:

1. A new style of programming which allows applications to be developed much faster than traditional methods allow, and makes it possible for applications to evolve rapidly to meet changing user requirements.This style of programming frequently uses a scripting language tointerconnect ‘off the shelf ‘components that are themselves written in conventional language. Applications built in this way are called ‘glue applications’, and the language is called a ‘glue language’.
2. A glue language is a programming language (usually an interpreted scripting language) that is designed or suited for writing glue code – code to connect software components. They are especially useful for writing and maintaining:

* Custom commands for a command shell
* Smaller programs than those that are better implemented in a compiled language
* "Wrapper" programs for executables, like a batch file that moves or manipulates files and does other things with the operating system before or after running an application like a word processor, spreadsheet, data base, assembler, compiler, etc.
* Scripts that may change
* Rapid prototypesof a solution eventually implemented in another, usually compiled, language.

Glue language examples:

AppleScript m4

ColdFusion Perl

DCL PHP

Python Erlang

Rebol JCL Rexx JScript

JavaScript Ruby

Lua Scheme  
Tcl Unix Shell  
scripts (ksh, csh,bash, shand others) VBScript

1. Using a scripting language to ‘manipulate, customize and automate the facilities of an existing system’, as the ECMAScript definition puts it. Here the script is used to control an application that provides a programmable interface: this may be an API, though more commonly the application is constructed from a collection of objects whose properties and methods are exposed to the scripting language.Example: use of Visual Basic for applications to control the applications in the Microsoft Office Suite.
2. Using a scripting language with its rich functionality and ease of use as an alternate to a conventional language for general programming tasks, particularly system programming and administration. Examples: are UNIX system administrators has for a long time used scripting languages for system maintenance tasks,and administrators of WINDOWS NT systems are adopting a scripting language ,PERL for their work.

Scripting is a part of UNIX cultural tradition and UNIX was the initial development platform for most of mainstream scripting languages with the exception of REXX. But they are portable and now all can be used in Windows and other OSes.

In the dawn of the Internet revolution, [web pages](https://www.ntchosting.com/encyclopedia/internet/web-page/) were HTML based – they looked good, but could not provide the rich functionality that we are used to today. This is the time when scripting languages first came to the Internet world. Scripts by then had been widely used in almost every computer in the world as an integral part of its operating system, but not on the [Internet](https://www.ntchosting.com/internet/) scene. And with the introduction of the scripting languages, the websites slowly began to enhance their functionality. And thanks to the use of scripts, we can have content- and functionality-rich [websites](https://www.ntchosting.com/encyclopedia/internet/website/).

* There are many scripting languages:  
  – Python, perl, R, javascript, PHP, ...
* Some scripting languages are used primarily in web systems. – Some are self-standing languages. However, others are embedded in HTML and used to enhance web pages.

**Characteristics of scripting languages:**

These are some properties of scripting languages which differentiate SL from programming languages.

Integrated compile and run: SL’s are usually characterized as interpreted languages, but this is just an oversimplification.They operate on an immediate execution, without need to issue separate command to compile the program and then to run the resulting object file, and without the need to link extensive libraries into the object code.This is done automatically.A few SL’S are indeed implemented as strict interpreters.

Low overheads and ease of use:

1. Variables can be declared by use
2. The number of different data types is usually limited
3. Everything is string by context it will be converted as number(vice versa)
4. Number of data structures is limited(arrays)

Enhanced functionality: SL’s usually have enhanced functionality in some areas. For example, most languages provide string manipulation based on the use of regular expressions, while other languages provide easy access to low-level operating system facilities, or to the API, or object exported by an application.

Efficiency is not an issue: ease of use is achieved at the expense of efficiency, because efficiency is not an issue in the applications for which SL’S are designed.

A scripting language is usually interpreted from source code or bytecode. By contrast, the software environment the scripts are written for is typically written in a compiled language and distributed in machine code form.

Scripting languages may be designed for use by end users of a program – end-user development – or may be only for internal use by developers, so they can write portions of the program in the scripting language.

Scripting languages typically use abstraction, a form of information hiding, to spare users the details of internal variable types, data storage, and memory management.

Scripts are often created or modified by the person executing them, but they are also often distributed, such as when large portions of games are written in a scripting language.

The characteristics of ease of use, particularly the lack of an explicit compile-link-load sequence, are sometimes taken as the sole definition of a scripting language.

* Both Batch and Interactive use.
* Economy of Expression.
* Lack of declarations; simple scoping rules.
* Flexible dynamic typing.
* Easy access to other programs.
* Sophisticated Pattern matching.
* High-level data types.

**Users For Scripting Languages:**

Users are classified into two types

1. Modern applications
2. Traditional users

Modern applications of scripting languages are:

1. Visual scripting: A collection of visual objects is used to construct a graphical interface.This process of constructing a graphical interface is known as visual scripting. The properties of visual objects include text on button, background and foreground colours.These properties of objects can be changed by writing program in a suitable language.  
   The outstanding visual scripting system is visual basic.It is used to develop new applications.Visual scripting is also used to create enhanced web pages.
2. Scripting components:In scripting languages we use the idea to control the scriptable objects belonging to scripting architecture.Microsoft's visual basic and excel are the first applications that used the concept of scriptable objects.To support all the applications of Microsoft the concept of scriptable objects was developed.
3. Web scripting: web scripting is classified into three forms. They are processing forms, dynamic web pages, dynamically generating HTML.

Applications of traditional scripting languages are:

1. System administration,
2. Experimental programming,
3. Controlling applications.

Application areas:

Four main usage areas for scripting languages:

1. **Command scripting languages:** Command scripting languages are the oldest class of scripting languages. They appeared in 1960, when a need for programs and tasks control raised. The most known language from the first generation of such languages is JCL (Job Control Language), created for IBM OS/360 operating system. Modern examples of such languages include shell language, described above, and also text-processing languages, such as sed and awk. These languages were one of the first to directly include support for regular expression matching - a feature that later was included into more general-purpose languages, such as Perl.
2. **Application scripting languages:** Application scripting languages were developed in 1980s, in the era of personal computers, when such important applications as spreadsheets and database clients were introduced, and interactive session in front of the PC became the norm. One of the prime examples of these languages is Microsoft-created Visual Basic language, and especially its subset named Visual Basic for Applications, designed explicitly for office applications programming.
3. **Mark-up language:** Mark-up languages are a special case in the sense that they are not a real programming languages, but rather a set of special command words called ’tags’ used to mark-up parts of text documents, that are later used by special programs called processors, to do all kinds of transformations to the text, such as displaying it in a browser, or converting it to some other data format. The basic idea of mark-up languages is the separation of contents and structure, and also including formatting commands and interactive objects into the documents. The first mark-up language named GML (Generic Mark-up Language) was created in 1969 by IBM. In 1986, ISO created a standard called SGML, based on GML ideas.
4. **Universal scripting languages:** The languages that belong to this class are perhaps the most well-known. The very term”scripting languages” is associated with them. Most of these languages were originally created for the UNIX environment. The goals however were different. The Perl programming language was made for report generation, which is even reflected in its name (Practical Extraction and Report Language). It is commonly said that the primary reason for its enormous popularity is the ability to write simple and efficient CGI scripts for forming dynamic web pages with this language. Perl was there in the right place at the right time. The Python language was originally made as a tool for accessing system services of the experimental operating system Amoeba. Later it became a universal object-oriented scripting language. Implementations exist for the Java Virtual Machine and also for Microsoft Intermediate Language used on Microsoft .NET platform.  
   Unlike Perl and Python, which make it easy to write completely standalone programs, Tcl relies heavily on C and C++ extension modules.

**Advantages and Disadvantages of Scripting Languages**

Scripting languages are becoming more popular due to the emergence of web-based applications. The market for producing dynamic web content is now expanding extremely rapidly such that new scripting languages have been developed to allow users with little or no programming expertise to develop interactive web pages with minimum effort.

Also the increases in computer performance over the past few years has promoted a increase in the power and sophistication of scripting languages that, unlike conventional programming languages, can even have certain security features built-in.

Major advantages of scripting languages include:

* Easy to learn and use
* Minimum programming knowledge or experience required
* Allow complex tasks to be performed in relatively few steps
* Allow simple creation and editing in a variety of text editors
* Allow the addition of dynamic and interactive activities to web pages
* Editing and running code is fast.

The major disadvantage of scripting languages is that executable code can inadvertently be downloaded from a remote server to a web browser's machine, installed and run using the local browser's interpreter. This is easily done by visiting dubious web sites or downloading programs without valid authenticity. The user is probably unaware of anything devious occurring.

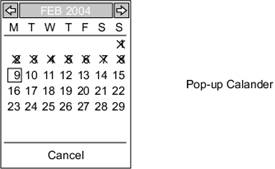
This is a weakness and major drawback in the formal rules defining scripting languages like JavaScript and VBScript.

**Uses of Scripting Languages**

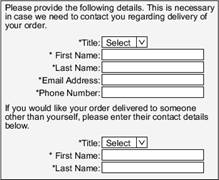
Typical uses of scripting languages including JavaScript include:

Image or text rollovers: If the user rolls the mouse over a graphic or hypertext then a text or graphic box will appear:

Creating a pop-up window to display information in a separate window from the web page that triggered it. This is useful if the user requires to perform a simple calculation or consult a calendar for inputting dates. This is achieved by embedding ActiveX controls or Java applets into the script.



Validating the content of fields: when filling in forms, each field, especially required fields denoted by an asterisk, are validated for correct input. If the field is left blank or incorrect information entered then a user message will be generated and you may not continue.



**Web scripting:**

Web is the most fertile areas for the application of scripting languages. Web scripting divides into three areas

1. Processing forms
2. Creating pages with enhanced visual effects and user interaction and
3. Generating pages ’on the fly’ from material held in database.

Processing Web forms:

In the original implementation of the web, when the form is submitted for processing, the information entered by the user is encoded and sent to the server for processing by a CGI script that generates an HTML page to be sent back to the Web browser.

This processing requires string manipulation to construct the HTML page that constitutes the replay, and may also require system access, to run other processes and to establish network connections. Perl is also a language that uses CGI scripting.

Alternatively for processing the form with script running on the server it possible to do some client –side processes within the browser to validate form data before sending it to the server by using JavaScript, VBScript etc.

Dynamic Web pages:

‘Dynamic HTML’ makes every component of a Web page (headings, anchors, tables etc.) a scriptable object. This makes it possible to provide simple interaction with the user using scripts written in JavaScript/Jscript or VBScript, which are interpreted by the browser.

Microsoft’s ActiveX technology allows the creation of pages with more elaborate user interaction by using embedded visual objects called ActiveX controls. These controls are scriptable objects, and can in fact be scripted in variety languages. This can be scripted by using Perl scripting engine.

Dynamically generated HTML:

Another form of dynamic Web page is one in which some or the entire HTML is generated by scripts executed on the server. A common application of the technique is to construct pages whose content is retrieved from a database. For example, Microsoft’s IIS web server implements Active Server Pages (ASP), which incorporate scripts in Jscript or VBScript.

**The universe of scripting languages:**

Scripting can be traditional or modern scripting, and Web scripting forms an important part of modern scripting. Scripting universe contains multiple overlapping worlds:

* The original UNIX world of traditional scripting using Perl and Tcl the Microsoft world of Visual Basic and Active controls.
* The world of VBA for scripting compound documents
* The world of client-side and server-side Web scripting.

The overlap is complex, for example web scripting can be done in VBScript, JavaScript/Jscript, Perl or Tcl. This universe has been enlarged as Perl and Tcl are used to implement complex applications for large organizations e.g Tcl has been used to develop a major banking system, and Perl has been used to implement an enterprise- wide document management system for a leading aerospace company.

**History of scripting languages:**

Scripting languages are not new: they have existed since at least the 1960s. However, the power and sophistication of scripting languages have improved dramatically in recent years. When coupled with tremendous increases in computer speed, it has become possible to use scripting languages for a much broader range of applications than was possible previously.

Nearly every major computing platform over the last four decades has supported both system programming languages (for creating applications from scratch) and scripting languages (for integrating components and applications). Here are a few examples of significant scripting languages, in rough chronological order.

**JCL (Job Control Language)**

Used to sequence job steps and arrange the flow of data in card decks for OS/360, starting in the 1960s. JCL may have been the first widely used scripting language, though it is a relatively primitive one by today's standards.

**Unix shells (sh, csh, ksh, ...)**

The first shell, sh, was written for the UNIX operating system in the early 1970's and a series of others followed over the next 20 years. Shell programs are used both for typing interactive commands and for writing scripts that automate common tasks. One of the most unique and powerful aspects of Unix was the ability to write shell scripts that create new applications by composing existing applications; it is perhaps the single most important reason for Unix's popularity as a platform for application developers.

**Rexx**

Michael Cowlishaw initially conceived the Rexx language in 1979 to simplify programming tasks on IBM's CMS timesharing system. It became popular as a macro language for arbitrary application programs, and its usage spread too many other platforms, including PCs and UNIX.

**Perl**

Created by Larry Wall in the late 1980s as a way to bring together in one place the functions of many popular UNIX text processing applications such as sh, sed, and awk, Perl quickly became a favourite tool of system administrators. With the arrival of the World-Wide Web, Perl achieved even greater fame as a convenient way to write CGI scripts for dynamic Web pages.

**Tcl**

Created by John Ousterhout in the late 1980s as an embeddable command language for interactive tools. When supplemented with the Tk toolkit, it became popular as the fastest way to build graphical user interfaces on UNIX. Tcl and Tk were ported to Windows and the Macintosh in the mid 1990s, producing an outstanding cross-platform development environment. Today Tcl is used for a wide variety of integration applications age1image37808including Web content generation, financial applications, electronic design automation, automated testing, and system management.

**Visual Basic**

This Microsoft product lies somewhere between a scripting language and a system programming language. It became popular in the early 1990s as the easiest way to create graphical user interfaces under Windows. The combination of Visual Basic and VBX (later ActiveX) components is probably the most successful component framework in existence, due in large part to the ease of integration provided by Visual Basic.

**Python**

Python is a dynamic object-oriented language, created by Guido van Rossum in the early 1990s to bridge the gap between shell and C programming. It’s elegant, easy to learn syntax, high level data types, elaborate library, portability, and ease of extending and embedding in C/C++ all contribute to its popularity. Originally designed as an advanced scripting language, it found new uses as a rapid application development language for web, database and GUI applications, as well as for distributed systems and mobile code.

**JavaScript**

Created in the mid 1990s by Netscape Corporation to perform scripting functions in Web browsers, such as simple form validation. JavaScript has become the de facto standard for client-side Web scripting, although it doesn't have much to do with Java.

**What’s the difference between Scripting and Programming Languages?**

Basically, all scripting languages are programming languages. The theoretical difference between the two is that scripting languages do not require the compilation step and are rather interpreted. For example, normally, a C program needs to be compiled before running whereas normally, a scripting language like JavaScript or PHP need not be compiled.  
   
Generally, compiled programs run faster than interpreted programs because they are first converted native machine code. Also, compilers read and analyze the code only once, and report the errors collectively that the code might have, but the interpreter will read and analyze the code statements each time it meets them and halts at that very instance if there is some error. In practice, the distinction between the two is getting blurred owing to improved computation capabilities of the modern hardware and advanced coding practices.  
   
Another point to be noted is that while classifying a language as scripting language or programming language, the environment on which it would execute must be taken into consideration. The reason why this is important is that we can design an interpreter for C language and use it as a scripting language, and at the same time, we can design a compiler for JavaScript and use it as a programming language. A live example of this is V8, the JavaScript engine of Google Chrome, which compiles the JavaScript code into machine code, rather than interpreting it.  
   
Some scripting languages traditionally used without an explicit compilation step are JavaScript, PHP, Python, VBScript.  
   
Some programming languages traditionally used with an explicit compilation step are C, C++.

**Comparison between Programming Language and Scripting Language:**

|  |  |  |
| --- | --- | --- |
|  | **Programming Language** | **Scripting Language** |
| **Definition** | A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer. | A scripting language, script language or extension language is a programming language that allows control of one or more applications. |
| **Type** | Compiler-based language | Interpreter based language |
| **Usage** | Developing something from scratch | Used to combine existing components |
| **Interpretation** | Programmed languages are compiled into a more compact form that does not need to be interpreted by another application in the same way. The compiled result is stand-alone. | Scripted languages are interpreted within another program (like JavaScript is put within HTML and then interpreted by the browser). |
| **Running** | Run independent of an exterior (or parent) program | Run inside another program |
| **Design** | Designed to get full usage of a language | Designed to make coding fast and simple |
| **Conversion** | Converts the whole program into machine language in one shot | Converts high level instructions into machine language |
| **Creation** | Creates a .exe file | Does not create a .exe file |
| **Compilation** | Needs to compile the program | There is no need to compile the program |
| **Coding** | Programming is making a full code of program | Scripts are just a piece of code |
| **Temperament** | Harder to code. Needs numerous lines of code for each function | Easier to code. Needs only a few short lines of code for each function |
| **Complexity** | Are complex | Are easy to use and easy to write |
| **Development Time** | Takes longer to develop as more code needs to be written. | Takes less time to code as it needs less coding. |
| **Support** | * Explicit support of Data Types * Rich support for User Interface Design * Rich support for Graphic Design | * Implicit support of Data Types * Limited support for User Interface Design * Limited or no support for Graphic Design |
| **Hosting** | Does not require a host. Is self-executable | Requires a host |
| **Cost** | Increased maintenance cost | Reduced maintenance cost |
| **Examples** | C, C++, C#, Java, VC++, VB, Basic, COBOL, Pascal, etc. | JavaScript, VB Script, Shell, Perl, Python, Ruby, Rexx, PHP, GameMonkey, Lua, etc. |

**Compiled Languages vs. Scripting Languages**

**Compiled Languages**

Computers do not actually understand the code we write. We need to translate our human-readable code to machine-readable code. Our code need to be translated into bits and bytes that can be executed by computers. This translation process is called compilation. Languages that require compilation are called compiled languages. C, C++ and Java are compiled languages.

**Scripting Languages**

Languages that not require compilation are called scripting languages. Perl, PHP, Python and Ruby are scripting languages. Those languages rely on our source-code all the time. Scripting languages didn’t have compiler or a compilation process. Those languages used interpreters to translate our source-code to machine executable code on the fly.

Mostly, we have been using compiled languages like C, C++, Java and Visual Basic. Java has an interpreter but it need to compile first. Earlier versions of Visual Basic were interpreted. Visual Basic 5 and later came with compiler. So, Java and Visual Basic should be called as compiled languages.

Imagine; you wrote an application with Java. Then you compile your application with javac. After compiling completed, you run your application. When running your application, you notice a bug. To fix it, you have to stop your application, go back to source code, fix the bug, wait for the code to recompile, and restart your application to confirm that it is fixed. And if you find another bug, you’ll need to repeat that process again.

In a scripting language, you can fix the bug and just need to reload your application —no need to restart or recompile anymore. It’s as simple as that.

But, the scripting languages have a big performance problem. Translating on the fly can affect the application’s performance. Typically, the pre-compiled code is faster than on the fly translating. But, good news is, there are ways to speed up scripted languages, including techniques such as code caching and persistent interpreters.